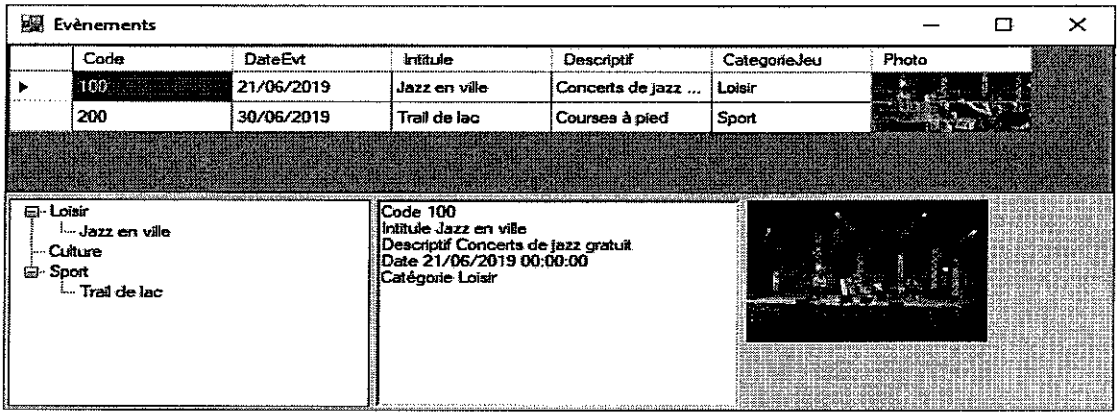
Licence Informatique **Lundi 17 juin 2019**
Conception et développement avancé d'applications **Examen - Durée 2h**
Documents CM, TD et TP autorisés

Exercice 1 : C# et Application Windows Forms (16 points)

Le conseil municipal d'une ville a décidé de demander le développement d'une application permettant de gérer les évènements qui se déroulent dans la ville. Dans un premier temps, une application prototype a été développée, pour tester la modélisation et quelques fonctionnalités.
Le prototype a été développé en langage C# sous forme d'une application Windows Forms. L'interface de l'application est la suivante :



Elle comporte plusieurs zones :

- La 1ère zone en haut permet d'afficher sous forme d'une grille (DataGridView nommée Grille) les informations sur les évènements.
- La 2ème zone en bas à gauche contient un arborescence (TreeView nommé Arbre) qui permet de présenter les évènements selon leur catégorie (loisir, culture, sport, etc.)
- La 3ème zone en bas au milieu comporte une zone d'édition (RichTextBox nommée Edition) pour afficher les informations d'un évènement sous forme textuelle
- La 4ème zone en bas à droite comporte à gauche la photo (PictureBox nommé Photo) de l'évènement dont le détail textuel est affiché en bas au milieu.

Dans un 1er temps, l'application comporte uniquement une classe (nommée « Evenement ») qui modélise un évènement et une classe (nommée « Form1 ») pour l'application principale qui gère une liste d'évènements.

- La classe « **Evenement** » modélise un évènement avec son code, son intitulé, son descriptif, sa date, sa catégorie et une photo. L'annexe 1 donne une description partielle de cette classe.
- La classe principale appelée « **Form1** » gère l'ensemble des évènements à l'aide d'une liste. L'annexe 2 donne une description partielle de cette classe.

Exercice 1 : (5,5 pts) POO et la classe « Evenement » (décrite partiellement en Annexe 1)

1. **(1 pt)** Donner le code du constructeur par défaut de la classe « **Evenement** » qui initialise tous les attributs à des valeurs par défaut (au choix) et en supposant qu'il existe une image nommée « PhotoDefaut » en ressources dans l'application pour initialiser la photo par défaut.
2. **(1,5 pts)** Compléter la classe « Evenement» pour qu'elle implémente l'interface « **IEquatable<T>** » qui ne comporte qu'une méthode « **public bool Equals (T other)** » en expliquant le code ajouté. La comparaison des évènements est faite sur leur « intitulé » uniquement.
3. **(3 pts)** Expliquer les trois principes fondamentaux de la POO cités ci-dessous en illustrant vos explications en définissant partiellement une classe « EvenementSponsor » qui est un événement qui comporte un ou des sponsors.
    - L'encapsulation et l'héritage
    - Le polymorphisme

Exercice 2 : (11,5 pts) La classe « Form1 » (décrite partiellement en Annexe 2)

Note : Seules les méthodes proposées dans le code des annexes sont à utiliser. Si vous jugez nécessaire d'utiliser d'autres méthodes, décrire leur code complet en C#.

1. **(1,5 pts)** Expliquer quels composants conteneurs ont été utilisés selon vous, pour construire l'interface décrite ci-dessus. Indiquer brièvement leur organisation sous une forme au choix (phrases ou schéma).

La liste des évènements est gérée dans une liste générique de type List<Evenement>.
Lors du lancement de l'application, plusieurs méthodes d'initialisation « InitListeEvts », « InitArbre », et « InitGrille » sont appelées dans le constructeur pour « remplir » les composants. La méthode « InitListeEvts » permet de créer des évènements pour tester l'application.
L'annexe 2 décrit partiellement la classe « Form1 ». En utilisant cette annexe, répondre aux questions suivantes.

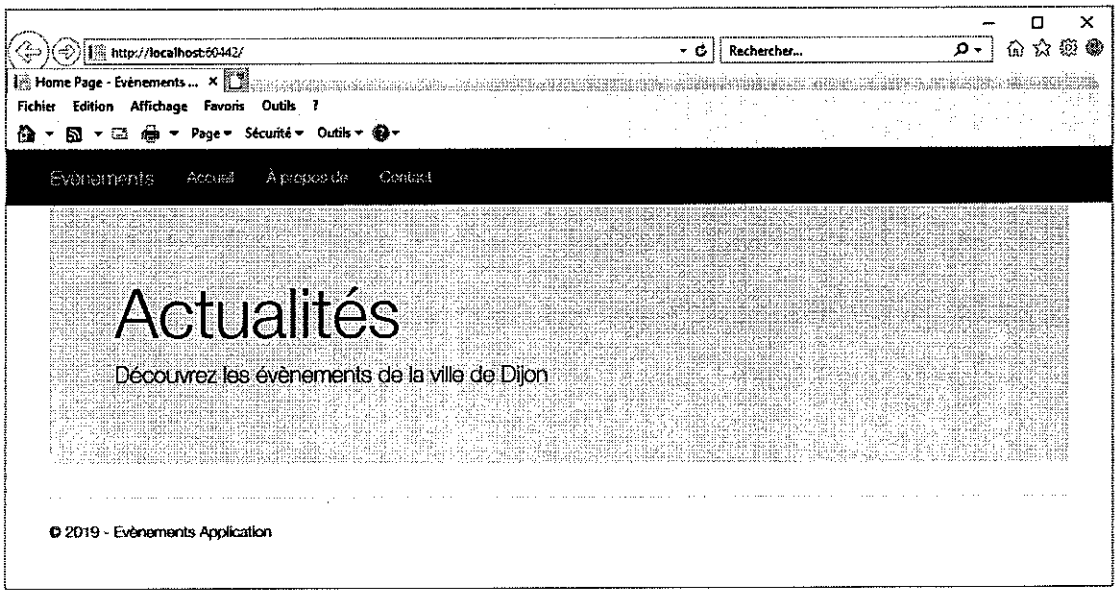
2. **(1,5 pts)** Donner le code de la méthode « InitListeEvts » pour ajouter l'évènement « Jazz en ville » décrit dans l'interface ci-dessus. Le nom de l'image en ressource correspondant à la photo est « Jazz ».
3. **(1,5 pts)** Compléter le code de la méthode « InitGrille » qui remplit la grille (DataGridView nommée Grille) avec toutes les informations des évènements.
4. **(2,5 pts)** Donner le code de la méthode « InitArbre » qui remplit le composant de type TreeView nommé Arbre, avec les intitulés des évènements classés par catégorie. On conservera pour chaque nœud le code de l'évènement dans sa propriété « Name ».
   On rappelle si besoin que l'instruction « int i = (int) o.Categorie; » permet de connaître l'indice de la catégorie dans le type énuméré, 0 pour le 1er, 1 pour le 2ème, etc.

   Le clic dans une cellule de la grille ou la sélection d'un intitulé d'évènement dans la l'arbre permet l'affichage des informations textuelles de l'évènement dans la zone d'édition (de type RichTextBox nommée Edition) et de sa photo dans le PictureBox nommé Photo.
5. **(2,5 pts)** Donner le code C# du gestionnaire de clic dans une cellule de la grille et expliquer ses paramètres.
   « private void Grille_CellClick(object sender, DataGridViewCellEventArgs e) »
6. **(2 pts)** Donner le code C# du gestionnaire de clic sur un nœud de l'arbre
   « private void ArbreCat_AfterSelect(object sender, TreeViewEventArgs e) »

Exercice 3 : Application APS.NET MVC (3 pts)

Afin de rendre l'application de gestion des évènements plus facilement accessible, il est envisagé de l'implémenter sous forme d'application Web ASP.net selon le modèle MVC en utilisant l'environnement Visual Studio 2015.
Cet environnement permet la génération d'un squelette d'application qui comporte une barre de menu avec 3 options : « Accueil » « A propos de » et « Contacts ». Ce squelette a été légèrement modifié pour être ensuite complété avec de nouvelles fonctionnalités.
Les annexes 3, 4 et 5 donnent le code des fichiers « Layout.cshtml », « HomeController.cs » et «Index.cshtml » générés dans ce squelette.

1. **(3 pts)** Expliquer le modèle d'architecture « Modèle – Vue – Controller » en détaillant le fonctionnement de la gestion du clic sur l'option « Accueil » du squelette de l'application.
   On précisera clairement sur cet exemple les parties de code concernées, la façon dont les événements sont déclenchés, les informations passées, etc.
   Vous pouvez utiliser les annexes pour les annoter si vous le souhaitez et les rendre avec votre copie.



ANNEXE 1 – Classe « Evenement »

```
public enum Categorie { Loisir, Culture, Sport} ;
```

```
public class Evenement
    {
       private int code;
       private string intitule;
       private string descriptif;
       private DateTime dt;
       private Categorie cat;
       private Image photo;

       public Evenement() { . . . }

       public Evenement(int c, string n, string de, DateTime d, Categorie ca, Image p)) { . . . }

       public int Code { get { return this.code; } }
       public DateTime DateEvt { get { return this.dt; } set { this.dt = value; } }
       public string Intitule { get { return this.intitule; } set { this.intitule = value; } }
       public string Descriptif { get { return this.descriptif; } }
       public Categorie CategorieJeu { get { return this.cat; } }
       public Image Photo { get { return this.photo; } set { this.photo = value; }
       public DateTime DateCreation
        { get { return this.dateCreation; } set { this.dateCreation = value; } }

        public override string ToString()
        {
            string s = "";
                s += "Code " + this.code + "\n";
                s += "Intitule " + this.intitule + "\n";
                s += "Descriptif " + this.descriptif + "\n";
                s += "Date " + this.dt.ToString() + "\n";
                s += "Catégorie " + Enum.Format(typeof(Categorie), cat, "g") + "\n";
                return s;        }

       . . .
}
```

ANNEXE 2 – Classe « Form1»

```
    public partial class Form1 : Form
    {    private List<Evenement> lst;
         public Form1()
         {
       InitializeComponent();
       This.lst = new List<Evenement>();
       InitListeEvts();
       Grille.DataSource = lst;
       InitArbre();
   }

   public Evenement GetEvtCode(int c)
   {   foreach (Evenement e in lst)
             if (e.Code == c) return e;
       return null; }

       public void InitListeEvts() { // A COMPLETER }

       public void InitGrille() { // A COMPLETER }

       public void InitArbre() { // A COMPLETER }

       private void Arbre_AfterSelect(object sender, TreeViewEventArgs e) { // A COMPLETER }

       private void Grille_CellClick(object sender, DataGridViewCellEventArgs e) { // A COMPLETER }
```

ANNEXE 3 – Application Web ASP.net - modèle MVC
Fichier modifié « Layout.cshtml » du squelette d'application

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>@ViewBag.Title - Evènements de la ville de DIJON</title>
    @Styles.Render("~/Content/css")
    @Scripts.Render("~/bundles/modernizr")

</head>
<body>
    <div class="navbar navbar-inverse navbar-fixed-top">
        <div class="container">
            <div class="navbar-header">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-
target=".navbar-collapse">
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                @Html.ActionLink("Evènements", "Index", "Home", new { area = "" }, new { @class =
"navbar-brand" })
            </div>
            <div class="navbar-collapse collapse">
                <ul class="nav navbar-nav">
                    <li>@Html.ActionLink("Accueil", "Index", "Home")</li>
                    <li>@Html.ActionLink("À propos de", "About", "Home")</li>
                    <li>@Html.ActionLink("Contact", "Contact", "Home")</li>
                </ul>
               <!-- @Html.Partial("_LoginPartial")-->
            </div>
        </div>
    </div>
    <div class="container body-content">
        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - Evènements Application</p>
        </footer>
    </div>

    @Scripts.Render("~/bundles/jquery")
    @Scripts.Render("~/bundles/bootstrap")
    @RenderSection("scripts", required: false)
</body>
</html>
```

ANNEXE 4 – Application Web ASP.net - modèle MVC
Fichier « HomeController.cs » modifié du squelette d'application

```
public class HomeController : Controller
    {
        public ActionResult Index()
        {     return View(); }

        public ActionResult About()
        {     ViewBag.Message = "Actualités";
              return View(); }

        public ActionResult Contact()
        {     ViewBag.Message = "Dijon contact";
            return View();
        }
     }
```

ANNEXE 5 – Application Web ASP.net - modèle MVC
Fichier « Index.cshtml » modifié du squelette d'application

```
@{
    ViewBag.Title = "Home Page";
}

<div class="jumbotron">
    <h1>Actualités</h1>
    <p class="lead">Découvrez les évènements de la ville de Dijon</p>
</div>
```